LA-UR-21-27215
Approved for public release; distribution is unlimited.

Title: Using ROS for Robotic Control

Author(s): Buford, Christopher Ricardo
Boardman, Beth Leigh

Intended for: LANL Student Symposium

Issued: 2021-07-23

Los Alamos NATIONAL LABORATORY

# Using ROS for Robotic Control

Undergraduate Student: Chris Buford II
Mentor: Dr. Beth Boardman

August 3, 2021

# Project Description

The use of robots is important, to help prevent people from being exposed to hazardous materials within a glovebox. The research proposal for the summer was to make a pick and place demo for a seven degrees-of-freedom Motoman Sia5d robot. The Robot Operating System (ROS) was used and the knowledge of C++ and Linux had to be applied. The robot was used to pick up and sort four objects (cylinder, sphere, rectangular prism, and cone) within a virtual environment.

# Agenda

1. Motivation
2. Summer Project Introduction
3. Robot: Motoman Sia5d
4. Robot Demo
5. Important Pieces of Code

# Motivation

- Glovebox work for hazardous material
  - Punctured gloves
  - Hazardous materials

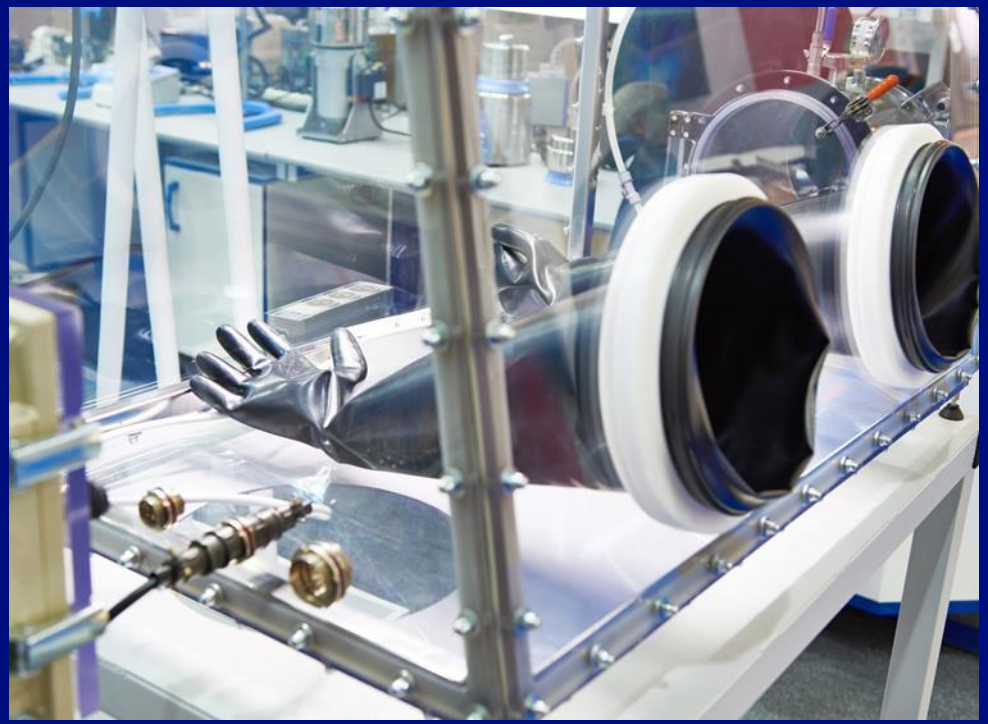


Figure 1: Glovebox

Figure 2: Another image of glovebox

# Summer Project Introduction

- Learn Robot Operating System (ROS)
  - C++ and Linux
  - Open-source framework
  - Build and use code for robotic applications

- Create demo to sort objects (in virtual environment)
  - Cylinder, sphere, cone, box

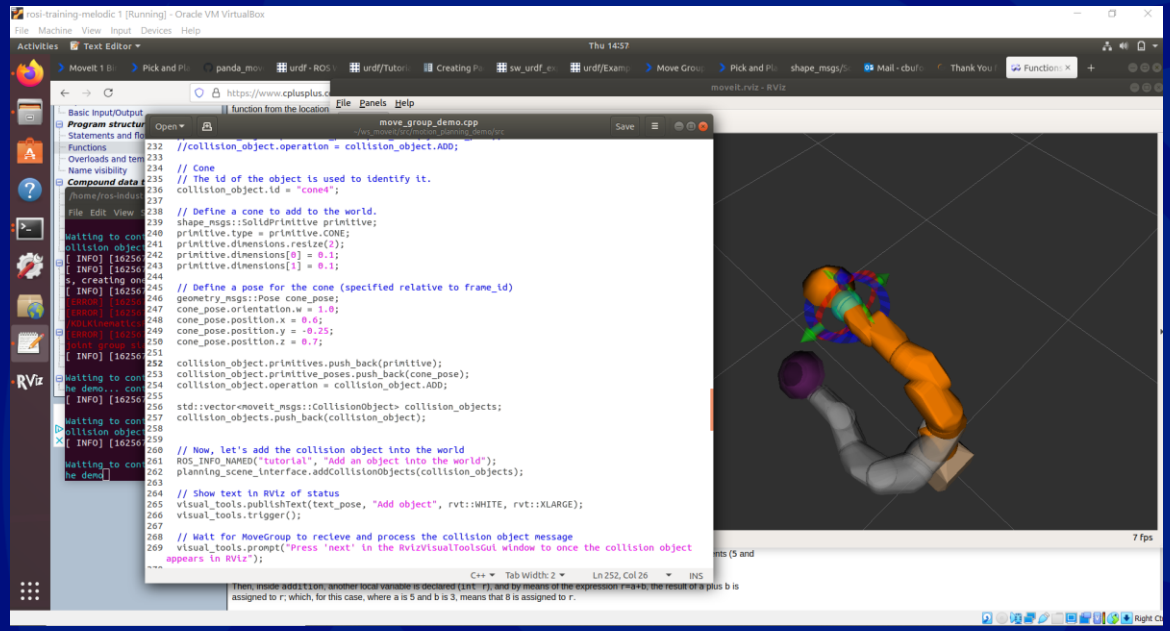



Figure 3: Rviz in ROS

# Robot: Motoman Sia5d

- 5.0kg Payload
- 559 mm Horizontal Reach
- 1007 mm Vertical Reach
- 7 degrees of freedom
- DX100 Controller

Figure 4: Motoman Sia5 Robot

# Details of Pick and Place Demo

1. Robot starts at Home Position
2. Add object into virtual space
3. Robot moves to Pick Position
4. Robot attaches object (picks up object)
5. Robot moves to Place Position
6. Robot detaches object (places object)
7. Robot moves to Home Position

- Repeats for each object

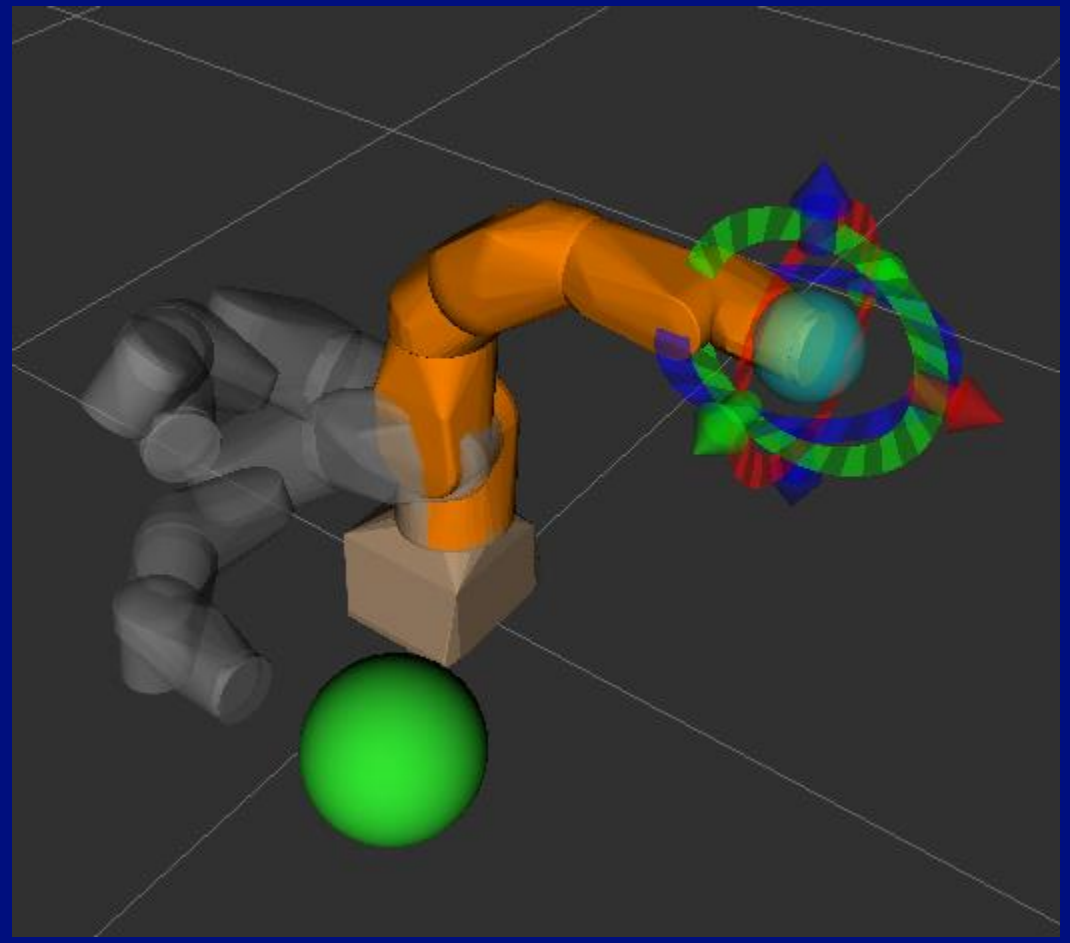
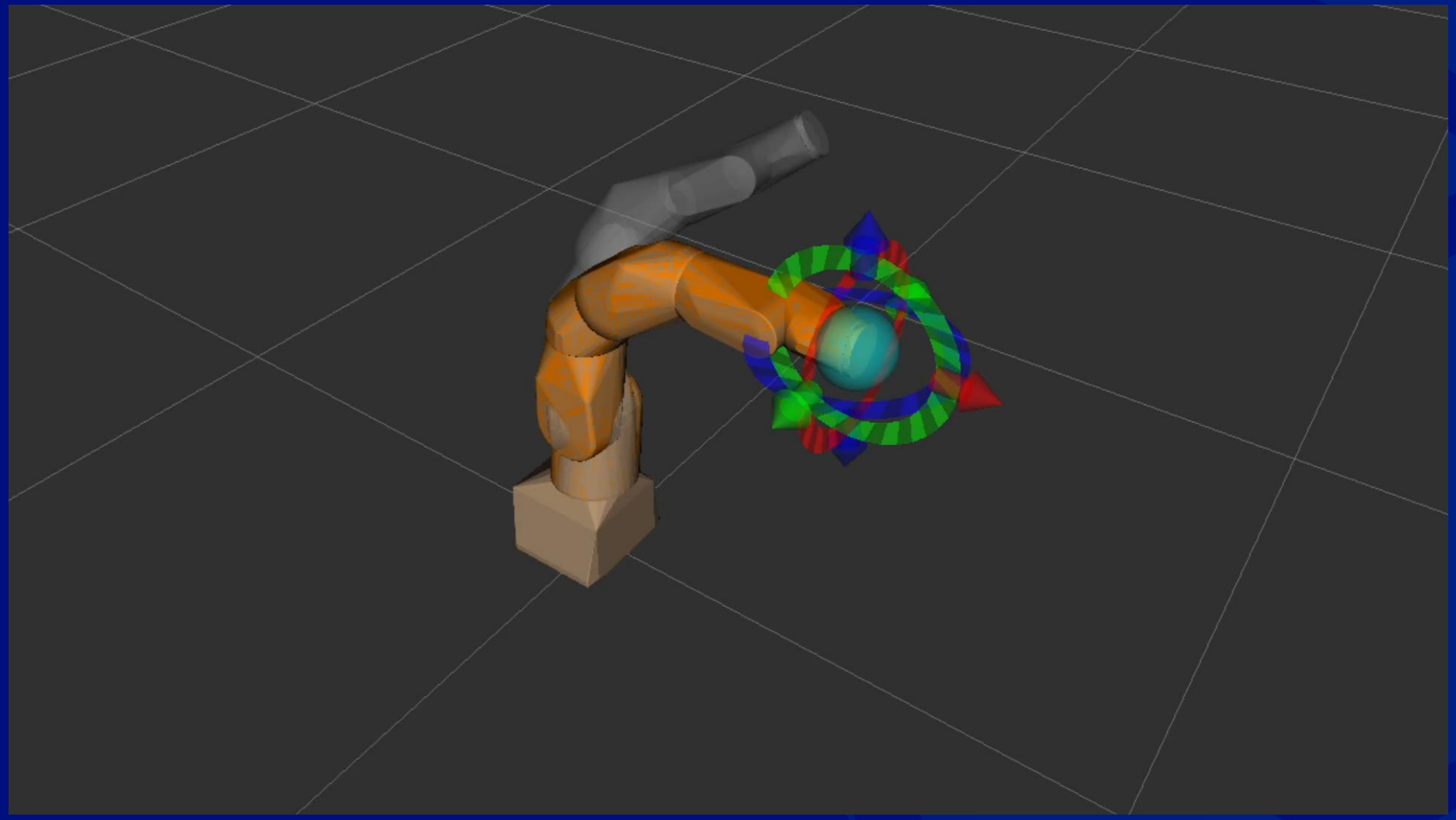


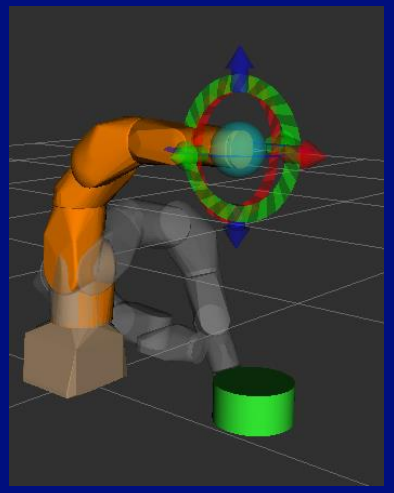
Figure 5: Robot picks up sphere

# Robot’s Pick and Place Demo (sphere)



# Pick and Place Position for Each Object

Figure 6a: Pick Position (cylinder)



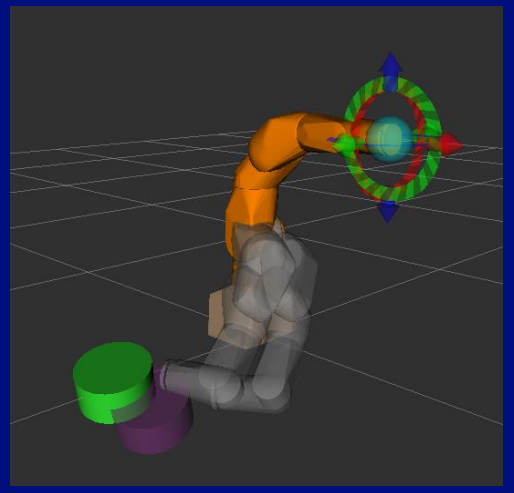
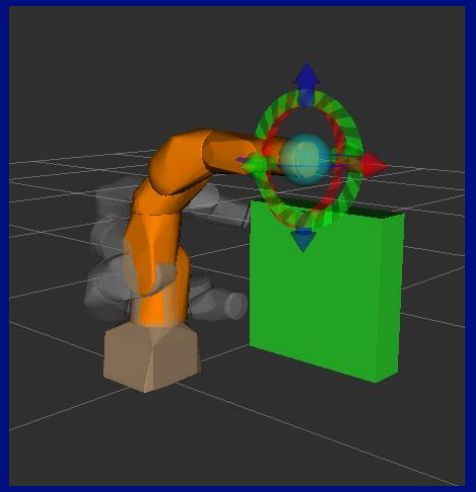


Figure 6b: Place Position (cylinder)

Figure 7a: Pick Position (box)



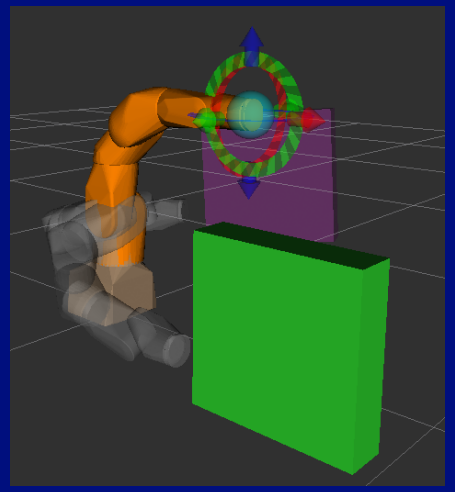
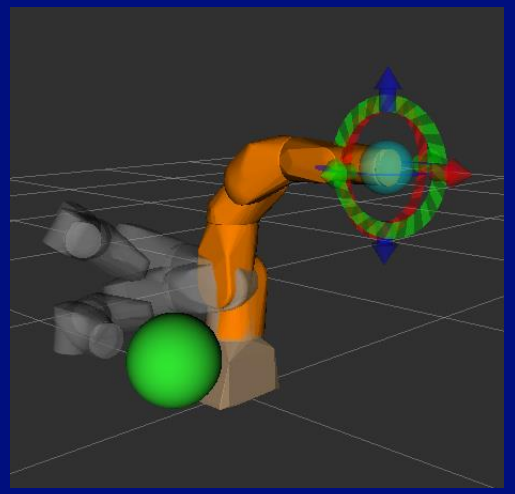


Figure 7b: Place Position (box)

# Pick and Place Position for Each Object

Figure 8a: Pick Position (sphere)



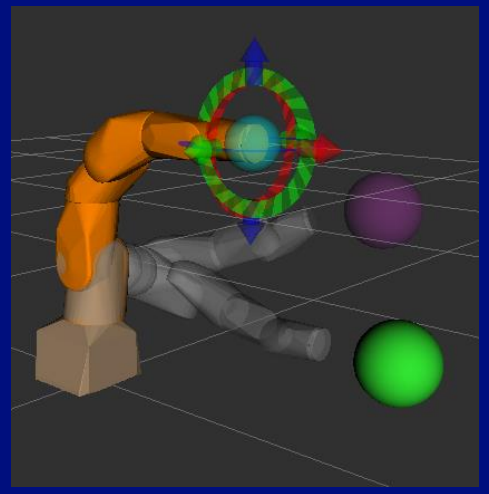
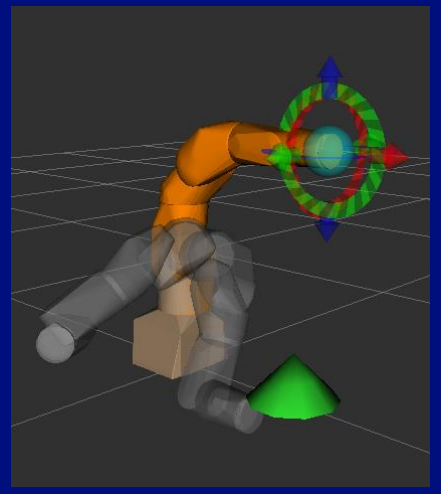


Figure 8b: Place Position (sphere)

Figure 9a: Pick Position (cone)



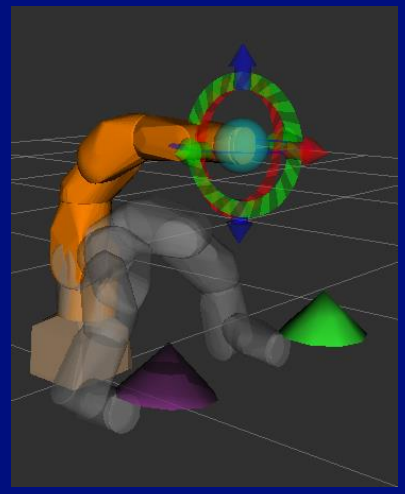


Figure 9b: Place Position (cone)

# Important Pieces of Code

- cd ~/ws_moveit/src
  - Cd means change directory
- catkin build
  - build one or more packages in a catkin workspace
- source ~/ws_moveit/devel/setup.bash
  - read and execute the content of a file

```
274   // Pick Position
275   // Next get the current set of joint values for the group.
276   std::vector<double> joint_group_positions;
277   current_state->copyJointGroupPositions(joint_model_group, joint_group_positions);
278
279   // Now, let's modify one of the joints, plan to the new joint space goal and
    visualize the plan.
280   joint_group_positions[0] = 1.0683981756808993;  // radians
281   joint_group_positions[1] = -1.320181899029397;  // radians
282   joint_group_positions[2] = -0.8069683938252558;  // radians
283   joint_group_positions[3] = -0.4665983158747622;  // radians
284   joint_group_positions[4] = 0.6910305310648733;  // radians
285   joint_group_positions[5] = 0.9785679348595806;  // radians
286   joint_group_positions[6] = 1.1235503435109293;  // radians
287   move_group.setJointValueTarget(joint_group_positions);
288
289   success = (move_group.plan(my_plan) ==
    moveit::planning_interface::MoveItErrorCode::SUCCESS);
290   ROS_INFO_NAMED("tutorial", "Visualizing pick position (joint space goal) %s",
    success ? "" : "FAILED");
291   move_group.move();
292
293   // Visualize the plan in RViz
294   visual_tools.deleteAllMarkers();
295   visual_tools.publishText(text_pose, "Joint Space Goal", rvt::WHITE, rvt::XLARGE);
296   visual_tools.publishTrajectoryLine(my_plan.trajectory_, joint_model_group);
297   visual_tools.trigger();
298   visual_tools.prompt("Press 'next' in the RvizVisualToolsGui window to continue the
    demo");
299
300   // Now, let's attach the collision object to the robot.
301   ROS_INFO_NAMED("tutorial", "Attach the object to the robot");
302   move_group.attachObject("cone4");
303
304   // Show text in RViz of status
305   visual_tools.publishText(text_pose, "Object attached to robot", rvt::WHITE,
    rvt::XLARGE);
306   visual_tools.trigger();
307
308   /* Wait for MoveGroup to recieve and process the attached collision object message
    */
309   visual_tools.prompt("Press 'next' in the RvizVisualToolsGui window to once the
    collision object attaches to the "
310                       "robot");
```

Figure 10: Robot code for Pick Position

# References


- *SIA5D*. Yaskawa. (n.d.). https://www.motoman.com/en-us/products/robots/industrial/assembly-handling/sia-series/sia5d.